PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 17/30

A1

(11) International Publication Number: WO 00/33216

(43) International Publication Date: 8 June 2000 (08.06.00)

(21) International Application Number: PCT/US99/28226

(22) International Filing Date: 29 November 1999 (29.11.99)

(30) Priority Data:

60/110,190	30 November 1998 (30.11.98)	US
60/163,345	3 November 1999 (03.11.99)	US
09/433,630	3 November 1999 (03.11.99)	US

(71) Applicant: LEXEME CORPORATION [US/US]; 56 John F. Kennedy Street, Cambridge, MA 02138 (US).

(72) Inventors: PUSTEJOVSKY, James, D.; 59 Claremont Avenue, Arlington, MA 02476 (US). CLIPPINGER, John, H.; Frank Kenison Road, Jefferson, NH 03583 (US). INGRIA, Robert; Apartment 4, 223 Broadway Street, Cambridge, MA 02139 (US).

(74) Agents: KANZAKI, Kim et al.; Townsend and Townsend and Crew LLP, 8th Floor, Two Embarcadero Center, San Francisco, CA 94111-3834 (US).

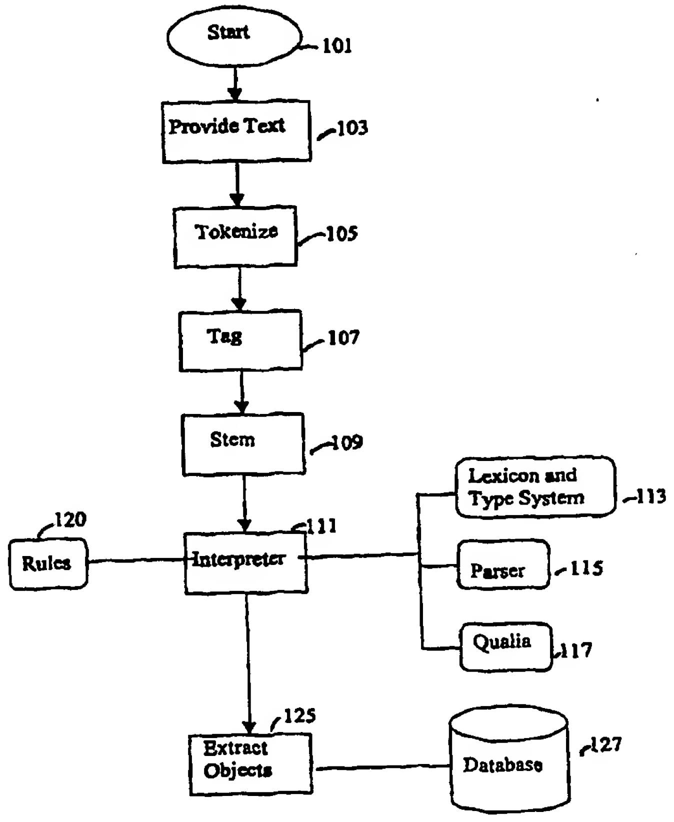
(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: A NATURAL KNOWLEDGE ACQUISITION METHOD

(57) Abstract

A natural language database forming method. The method includes providing text information (103) comprising a plurality of related words. A step of tagging (107) each word in the text information is also included. The method forms an object (125) that has syntactic information and semantic information from each word in the text information. The object is placed or mapped into an object oriented relational database (127).



AQ

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece	ML	Mali	TR	Turkey
BG	Bulgaria	HU	Hungary	MN	Mongolia	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MR	Mauritania	UA	Ukraine
BR	Brazil	IL	Israel	MW	Malawi	UG	Uganda
BY	Belarus	IS	Iceland	MX	Mexico	US	United States of America
CA	Canada	IT	Italy	NE	Niger	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NL	Netherlands	VN	Viet Nam
CG	Congo	KE	Kenya	NO	Norway	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NZ	New Zealand	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	PL	Poland		
CM	Cameroon	KR	Republic of Korea	PT	Portugal		
CN	China	KZ	Kazakstan	RO	Romania		
CU	Cuba	LC	Saint Lucia	RU	Russian Federation		
CZ	Czech Republic	LI	Liechtenstein	SD	Sudan		
DE	Germany	LK	Sri Lanka	SE	Sweden		
DK	Denmark	LR	Liberia	SG	Singapore		
EE	Estonia						

A NATURAL KNOWLEDGE ACQUISITION METHOD

CROSS-REFERENCES TO RELATED APPLICATIONS

This application claims priority from the following provisional patent application, the disclosure of which is herein incorporated by reference for all purposes:

U.S. Provisional Patent Application No. 60/110,190 in the names of James D. Pustejovsky, et al. titled,"Natural Knowledge Acquisition Method, System, and Code," filed November 30, 1998.

The following one commonly-owned co-pending provisional application is being filed concurrently and is hereby incorporated by reference in its entirety for all purposes:

U.S. Provisional Patent Application Serial No.,_____________, in the name of James D. Pustejovsky, titled, "A Method of Using a Natural Knowledge Acquisition System," (Attorney Docket Number 019497-000140)

BACKGROUND OF THE INVENTION

This invention generally relates to the field of information management. More particularly, the present invention provides a technique including a method for extraction and automatic classification of document content for any machine-readable text.

The expansion of the Internet has proliferated “on-line” textual information. Such on-line textual information includes newspapers, magazines, WebPages, email, advertisements, commercial publications, and the like in electronic form. By way of the Internet, millions if not billions of pieces of information can be accessed using simple “browser” programs. Information retrieval (herein “IR”) engines such as those made by companies such as Yahoo! allow a user to access such information using an indexing technique. The indexing technique includes full-text indexing, in which content words in a document are used as keywords. Full text searching had been one of the most promising of recent IR approaches. Unfortunately, full text searching has many limitations. For example, full text searching lacks precision and often retrieves literally thousands of "hits" or related documents, which then require further refinement and filtering. Additionally, full text searching has limited recall characteristics. Accordingly, full text searching has much room for improvement.

Techniques such as the use of "domain knowledge" can enhance an effectiveness of a full-text searching system. Domain knowledge techniques often provide related terms that can be used to refine the full-text searching process. That is, domain knowledge often can broaden, narrow, or refocus a query at retrieval time. Likewise, domain knowledge may be applied at indexing time to do word sense disambiguation or simple content analysis. Unfortunately, for many domains, such knowledge, even in the form of a thesaurus, is either generally not available, or is often incomplete with respect to the vocabulary of the texts indexed.

There have been attempts to use natural language understanding in some applications. As merely an example, U.S. Patent No. 5,794,050 in the names of Dahlgren et al. (herein Dahlgren.) utilized a conventional rule based system for providing searches on text information. Dahlgren, et al. use a naive semantic lexicon to "reason" about word senses. This simple semantic lexicon brings some "common sense" world knowledge to many stages of the natural language understanding process. Unfortunately, the design of such a semantic lexicon follows fairly standard taxonomic knowledge representation techniques, and hence the reasoning process making use of this taxonomy is generally incomplete. That is, it may provide a first level method for performing a relatively simple search, but often lacks a general ability to conduct a detailed retrieval to provide a comprehensive answer to a query. Fundamentally, the method and system described in Dahlgren, employs a natural language understanding system to provide a "concept annotation" of text for subsequent retrieval. Furthermore, when the system is used to query a database, it matches on pointers to the text provided by the annotation rather than an answer to the query.

Although some of the above techniques are fairly sophisticated compared to the information retrieval search engines so ubiquitous on the internet (e.g., Inktomi or Alta Vista), the results of the queries are "hits" rather than "answers"; that is, a hit is the entire text that matches the indexing criteria, while an answer on the other hand is the actual utterance (or portion of the text) that satisfied a user query. For example, if the query were "Who are the officers of Microsoft, Inc?", a hit-based system would return all the documents that contain this information anywhere within them, whereas an answer-based system would return the actual value of the answer, namely the officers.

From the above, it is seen that a technique for improved information retrieval is highly desirable.

SUMMARY OF THE INVENTION

According to the present invention, a technique including a method for acquiring information is provided. In a specific embodiment, the present invention provides a method using a combination of syntactic and semantic information objects.

In a specific embodiment, the present invention provides a natural language database forming method. The method includes providing text information comprising a plurality of related words. A step of tagging each word in the text information also is included. The method forms an object that has syntactic information and semantic information from each word in the text information. The object is placed or mapped into a object oriented relational database.

In an alternative embodiment, the present invention provides a natural language knowledge acquisition method. The method includes providing text information (e.g., electronic form) including a plurality of related words. The method tags each word in the text information. The method also forms an object comprising syntactic information and semantic information from each word in the text information. The object is placed into a relational, object-oriented, or mixed relational/object oriented database. The methods of providing, tagging, forming, and placing are repeated to populate the database. Next, a user can access the information in the database. Here, the user forms a query, which is entered, processed by the system, and selects an object based upon entity relationships to achieve a unique output, which can actually be an answer to the query.

In another specific embodiment, the present invention provides a method for recognizing lexical objects within text and typing these lexical objects into semantic categories. These semantic representations can then be utilized in a variety of ways, including, persisting them in various forms (such as relational, object, or mixed object/relational databases), text summarization, keyword extraction, semantic indexing.

Moreover, this method of deriving semantic representations from lexical objects in input text can be used both for extracting information and for querying an already existing database of knowledge (including those created by this engine). Thus both database population and database retrieval of said objects may be performed.

Numerous advantages are achieved by way of the present invention. In one embodiment, the present invention provides a relational database that can be queried using a natural language approach. In other aspects, the present invention provides methods using a combination of data coupled with logic. The invention can also provide knowledge extraction in other embodiments. The invention provides object creation, and provides conversational access to the database in other embodiments. Accordingly, the present invention can provide an acquisition technique that actually provides answers to queries (rather than hits), which can be singular. Depending upon the embodiment, one or more of these advantages can be present. These and other advantages, however, are described throughout the present specification and more particularly below.

These and other embodiments of the present invention are described in more detail in conjunction with the text below and attached Figs.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a simplified diagram of an information acquisition method according to an embodiment of the present invention; and

Fig. 2 is a simplified diagram of an information acquisition method according to an alternative embodiment of the present invention

DESCRIPTION OF THE SPECIFIC EMBODIMENTS

According to the present invention, a technique including a method for acquiring information is provided. In a specific embodiment, the present invention provides a method using a combination of syntactic and semantic information objects. In one or more aspects, the present invention provides a modular, object-oriented, and collaborative approach to semantic typing, interpreting, and extraction, of knowledge objects from text sources into databases. The invention provides a highly general object-oriented method for using lexically-based knowledge to identify and extract semantic objects in text and to represent them in a database. The invention offers savings in the time, effort, and costs for constructing, populating, and updating a wide variety of databases.

A method according the present invention is briefly outlined below:

1. Providing text information sources;
2. Tokenizing;
3. Performing part-of-speech tagging;
4. Stemming tagged items;
5. Interpreting including type composition and type induction, semantic-syntactic composition, and instantiation of qualia.
6. Translating the resulting expression into a relational model;
7. Inserting this into a relational database;
8. Performing other steps, as desired.

The above sequence of steps generally provides a method for natural language input into a relational database. The present sequence of steps using a combination of syntactic and semantic information using an object-oriented approach. Step 7 can use either a relational, object-oriented, or mixed relational/object database. If a relational database is used, the step 7 of translation into the relational model is required (step 6). Any relational database can be used in Step 7. As merely an example, the database can be made by a company called Oracle of Redwood City, California. Alternatively, other companies such as Informix, Sybase, and others also manufacture database designs that can incorporate the present invention. Similarly, any object-oriented or mixed relational/object database can be used. Details of the above steps are briefly described according to Fig. 1, for example.

Fig. 1 is a simplified diagram 100 of an information acquisition method according to an embodiment of the present invention. This diagram is merely an example and should not limit the scope of the claims herein. One of ordinary skill in the art would recognize other variations, modifications, and alternatives. The method 100 begins at start, step 101, which includes steps of providing text information (step 103), tokenizing (step 105), tagging (step 107), stemming (step 109), interpreting (step 111), and extracting objects (step 125). Further details of each of these steps are provided below.

The method provides (step 103) information such as text information from a variety of sources, including all digital sources. The sources include, among others, newspapers, magazines, research, web sites, product information, internets, ,intranets, and spoken language inputs. The text source are generally in electronic form, which can be read, organized, and categorized by way of a computer. The text source can be in any suitable electronic form such as ASCII, HTML, XML, LaTeX, word processing applications, and presentation slides (e.g., Microsoft Power Point™).

In the next step 105, the method tokenizes the text information. The text may be "tokenized," for example, split up into textual elements separated by a delimiter, such as a "white space" or "blank" character. Tokenization normalizes the input text into a form that is usable by subsequent steps of the method. In a specific embodiment, the tokenizer separates punctuation (e.g., periods, apostrophes, quotes, etc.) from words. As merely an example, the following phrase:

"Thomas E. Wheeler, CTIA's President"

is converted into:

" Thomas_E._Wheeler_,_CTIA_'s_President_"

where the underscore character "_" is used to represent a blank space for readability purposes only.

Other examples of tokenization are expansions of contractions:

he'd	==> he_'d
I'll	==> I_'ll

Other examples are shown below:

.... business day.	==> ..._business_day_.
.... reason to be cheerful:	==> ..._reason_to_be_cheerful_:
"They are in trouble."	==> "_They_are_in_trouble_. "

where again the underscore character "_" is used to represent a blank space for readability purposes only.

It is, however, desirable that abbreviations and initials be preserved and not separately tokenized, i.e., the period is not separated from them. For example, Mr., M.D., Mrs., Esq., and so on. It is also important not to split names that have false punctuation in them; e.g., index.html, http://www.company.com.

In a specific embodiment, the present invention provides a step of tagging (step 107) the tokenized text information. In a specific embodiment, a part-of-speech (herein "POS") tagger can be used. A goal of the part-of-speech tagger is to assign grammatical category labels to each tokenized element in the text produced from the tokenizer. For example, a tagger will convert the input below:

The_new_company_.

into

The/DT_new/JJ_company/NN_./.

where the underscore character "_" is again used to represent a blank space for readability purposes only and where the "/tag" labels, e.g., /DT, /JJ, /NN, refer to a standard set of POS tags used in the computational linguistics community. An example of such a POS tagging system is described in Brill (Brill, "AEric, "A simple rule-based part-of-speech ".tagger". In *Third Conference on Applied Natural Language Processing*, pages 152-155, Trento, Italy, 1992, which is herein incorporated by reference). An example of such a POS tagging system is described in Brill (Brill, Eric, Third Conference on Applied Natural Language Processing, pages 152-155, Trento, Italy, 1992).

The present method performs a step 109 of stemming. Stemming is yet another stage in normalization for further processing. For example, all stems are orthographically lower case. In addition, for example, in the case of inflected categories, such as a plural noun or a past tense verb, the stem will be the dictionary look up form of the token (e.g., 'man' for 'men', 'run' for 'ran'). Stemming can use dictionary lookup in the case of known inflected words. If the particular token does not occur in the dictionary, then it can be passed on to a stripped down version of the Porter Stemmer (Porter, M.F., "An Algorithm for Suffix Stripping," Program 14 (3), July 1980, pp. 130-137, which is herein incorporated by reference), which strips off affixes in certain orthographic contexts.

As merely an example, which should not limit the scope of the present invention herein, an illustration of steps of tokenizing, tagging, and stemming a text taken from a newspaper corpus is shown below.

Original Text:

Clinton Picks General to Command NATO WASHINGTON (Reuter) - President Clinton has chosen U.S. Army Gen. Wesley Clark to become commander of all allied NATO forces and American troops in Europe, a senior Pentagon official said Monday. Clark , 52, speaks Russian and was a member of the American team that helped broker the 1995 Dayton peace accords on Bosnia . He is based in Panama as chief of U.S. forces in Latin America and would replace retiring U.S. Army Gen. George Joulwan as Supreme Allied Commander of NATO in Europe (SACEUR) based in Mons, Belgium.

Examples of tokenizing, tagging and stemming of the above text:

Clinton/NNP Picks/VBZ General/NNP to/TO Command/NNP NATO/NNP WASHINGTON/NNP (/(Reuter/NNP)/SYM -/: President/NNP Clinton/NNP has/VBZ chosen/VBN U.S./NNP Army/NNP Gen./NNP Wesley/NNP Clark/NNP to/TO become/VB commander/NN of/IN all/DT allied/VBN NATO/NNP forces/NNS and/CC American/JJ troops/NNS in/IN Europe/NNP ,/, a/DT senior/JJ Pentagon/NNP official/NN said/VBD Monday/NNP ./. Clark/NNP ,/, 52/CD ,/, speaks/VBZ Russian/NNP and/CC was/VBD a/DT member/NN of/IN the/DT American/JJ team/NN that/WDT helped/VBD broker/NN the/DT 1995/CD Dayton/NNP peace/NN accords/NNS on/IN Bosnia/NNP ./. He/PRP is/VBZ based/VBN in/IN Panama/NNP as/IN chief/NN of/IN U.S./NNP forces/NNS in/IN Latin/NNP America/NNP and/CC would/MD replace/VB retiring/VBG U.S./NNP Army/NNP Gen/NNP ./. George/NNP Joulwan/NNP as/IN Supreme/NNP Allied/NNP Commander/NNP of/IN NATO/NNP in/IN Europe/NNP (/(SACEUR/NNP)/SYM based/VBN in/IN Mons/NNS ,/, Belgium/NNP ./.

Next, the method performs interpreting (step 111), using an interpreting module or the like. A specific embodiment of the interpreting, includes three sub-modules: accessing of a lexicon and type system 113, parsing 115, and identification of semantic types 117, e.g., qualia roles.

The first sub-module 113, accessing of a lexicon and type system 113, uses two knowledge bases. The first is a lexicon (resource A), indexed by stem within a particular part of speech. For example, 'base' as a noun and 'base' as a verb will have two separate entries. Each lexical entry contains a type property which is a name of a semantic type in the type system (resource B). Each lexical entry contains appropriate syntactic information. This information is combined to create the appropriate type of syntactic constituent (e.g., a noun for a stem with a noun tag), with the appropriate semantic representation.

Next, the parser sub-module 115 takes the output of the accessing of a lexicon and type system sub-module 113, and composes these into larger syntactic and semantic structures that make up the sentences of text in natural language. This sub-module 115 uses an engine embodying an all paths parser (Younger, D., "Recognition and Parsing of Context-free Languages in Time n^3", Information and Control 10:189-208, 1967, and Graham, Harrison, and Ruzzo, "An Improved Context-free Recongizer", ACM Transactions on Programming Languages and Systems 2:415-462, 1980, both of which are herein incorporated by reference), that uses a chart to hold information about the syntactic constituents found. Unlike a chart parser implemented in a procedural language, however, the present parser does not contain a single "controlling" module that acts in isolation to build interpreted structures out of passive data elements. Rather, interpreting can be accomplished only by collaboration among active objects of different classes.

As part of the Interpreting Step 111, syntactic-semantic composition is accomplished by means of grammar rules, 120, which specify both how syntactic elements are to be combined, and also how their semantic interpretations are to be composed. Moreover, grammar rules can contain constraints which specify the conditions under which a rule can apply. If the constraint fails, the rule is not even considered, which improves the performance of the interpreter, since useless search paths are not pursued.

An example of such a rule is

VP ==> VP NP {Transitive} [DirectObject]

This rule states that a VP (Verb Phrase) can be constructed from a VP followed by an NP (Noun Phrase). {Transitive} represents a constraint. This rule may only be fired if the VP is transitive: i.e., if the semantics of the VP allows for a direct object and this direct object position has not yet been filled. [DirectObject] represents a role, i.e., names of pieces of code that contain (1) constraints on the other dependent(s) of the rule; and (2) specify how the semantics of the dependents of the rule are combined to create the semantics of the dominating constituent (i.e. the left hand side). This is the semantic composition that takes place if the rule succeeds. In this case, the semantics of the NP is checked for compatibility with the semantic type specified for the direct object in the semantics of the VP. Moreover, if the type of the NP is compatible with the type of the direct object but is less specific, the type of the NP will be changed to the more specific type. It is in this way that the system acquires new knowledge, by using its existing knowledge bases to learn or further specify the meanings of words it has not previously encountered.

The semantics of the NP is then bound to the direct object position of the semantics of the existing VP, and this new semantic representation is made the semantics of the newly constructed VP.

In a specific embodiment, the present interpreter 111 uses a single Interpreter object. The interpreter 111 can manipulate a plurality of types of data structures such as those noted below:

1. The elements that represent completed syntactic elements (i.e., those that have been found);

2. The elements that represent incomplete syntactic elements (i.e. those that are in the process of being constructed); and

3. WH elements (i.e., words like 'who' and 'what' that appear 'dislocated' from the 'logical' syntactic positions in which they receive their semantic interpretation).

These types of syntactic elements are represented by objects of the class Edge. It is the interactions among the Edges that the Interpreter object "facilitates" in the construction of the larger syntactic structure: As active and inactive Edges are inserted into the chart representation that can be maintained by Interpreter, the Interpreter passes information about their presence to other Edges already in the chart that may be able to interact with them to create new Edges (i.e. to form new syntactic constituents). Similarly, the Interpreter passes information about Edges already in the chart (i.e.

syntactic constituents already found) to the new Edge that are added. The Interpreter also passes information about the existence of Edges representing WH elements to active Edges that may make use of them.

Edges interact to create new Edge on the basis of two other classes of objects: GrammarRule objects, which represent information about how new constituents can be built out of existing constituents, and Constituent objects, which represent traditional grammatical elements such as nouns, verbs, and sentences.

In a specific embodiment, the interpreting process can be defined as follows: the Interpreter is sent the output of the Stemmer, which is an Array of underspecified objects of type Constituent. Each of these objects includes the following pieces of information:

a). token: the unit of the input string (i.e. orthographic word or punctuation) found by the Tokenizer;
b). tag: the part of speech tag assigned by the Tagger;
c). stem: the dictionary lookup form of the token added by the Stemmer;
d). offset: the numerical position of the current token in the input text computed by the Stemmer.

The Interpreter object processes each of the underspecified Constituent objects in its input in sequence and tells it to transform itself into a fully specified object of the appropriate syntactic category. For example, a Constituent with token 'men' and tag 'NNS' can be transformed into an object of class Noun with the features (instance variables) 'proper = false' and 'number = plural'; a constituent with token 'sang' and tag 'VBD' is transformed into an object of class Verb with the feature 'tense = past'; etc.

As each input Constituent is processed, the Interpreter causes an associated Edge object to be created, which stores its paired constituent in an instance variable. In addition, the Interpreter consults the GrammarRule class object (step 120) to find out what grammar rules involve the new constituent. For each such grammar rule found, the Interpreter causes a new Edge to be built, which stores information about the rule that sanctioned its creation and the constituent that corresponds to the portion of the rule already found in the input.

To illustrate this process, let us look at what happens when the Interpreter object receives the output of the Stemmer for the input text 'green apples', given the existence of a GrammarRule object that constructs a NounPhrase out of an Adjective and a Noun. The Stemmer's output has two Constituents, the first with the token 'green' and tag 'JJ'; the second with the token 'apples' and tag 'NNS'. The Interpreter object processes these constituents in left to right order in a specific embodiment.

First it tells the Constituent for 'apple' to transform itself into a completely specified syntactic object: this produces an object of class Adjective with the feature 'degree = positive'. The Interpreter then causes an (inactive) Edge associated with this constituent to be built. It then checks to see if there are any existing Edges to the left of the Adjective that can use it, but since this is the first element in the sentence, there are none. Consulting with the GrammarRule class object, together they find the rule NounPhrase => Adjective Noun, which can make use of the Adjective object just created. The Interpreter, therefore, causes a new Edge to be built, which records that it has found the newly created Adjective object, and that it is trying to form a NounPhrase on the basis of the rule just given. The Interpreter then looks for any existing Edges (to the right of the Adjective), that the newly created Edge can interact with, but at this point there are none.

The Interpreter then tells the next Constituent object ('apples') to transform itself: this produces an object of class Noun with the features 'proper = false' and 'number = plural'. As before, the Interpreter causes a new associated Edge object to be built. It also looks for any (active) Edges immediately preceding 'apples'. It finds one, the active Edge built off of the Adjective object 'green', which is looking for an immediately following Noun. The Interpreter passes the new Edge to this existing Edge, which checks to see if it can use the newly formed Edge. It can, and therefore adds the new Noun to its collection of found constituent and marks itself as complete, i.e. as having found a NounPhrase. The Interpreter will also, as in the case of the preceding Adjective object, look for any new rules that can use the newly found Noun object and, if any such rules are found, will cause the corresponding Edges to be built, and will look for any Edges to the right of these new Edges that they can consume. These activities are irrelevant in the current example, however, so we omit the details.

In a specific embodiment, the present method using selected grammar rules (step 120) for a interpreter (step 111). The rules are noted below and are defined according to the following representations. The elements between curly brackets {} are constraints, i.e. names of pieces of code that the left edge of a rule (the left-most element on the right hand side) should satisfy before the rule will be considered. The elements between square brackets [] are roles, e.g., [DirectObject];

Utterance rules (8):

Utterance => RootS
Utterance => Interjection
Utterance => NP
Utterance => VP {Imperative} [ImperativeHead]
Utterance => RootS EndPunct [SemanticHead]
Utterance => Interjection EndPunct
Utterance => NP EndPunct [NPUtterance]
Utterance => VP EndPunct {Imperative} [ImperativeHead]

RootS rules (8):

RootS => WhPhrase RootS [WhQuestion]
RootS => NP VP [Subject]
RootS => V RootS {PossibleAuxiliary} [SubjectAuxInversion]
RootS => V NEG RootS {PossibleAuxiliary} [SubjectAuxInversion] [Negation]
RootS => V NP AdjP {Copula} [InvertedSubject] [InvertedPredicateAdjective]
RootS => Modal RootS [SubjectAuxInversion]
RootS => Modal NEG RootS [SubjectAuxInversion] [Negation]
RootS => AdvP RootS [AdverbialModifier]

<u>ComplementS rules (3):</u>

ComplementS => WhPhrase ComplementS [WhQuestion]

ComplementS => NP VP [Subject]

ComplementS => COMP ComplementS [Complementizer]

<u>RelS rules (1):</u>

RelS => WhRelPhrase ComplementS [RelativePronoun]

<u>ComplementVP rules (3):</u>

ComplementVP => WhPhrase ComplementVP [WhQuestion]

ComplementVP => COMP ComplementVP {InfinitivalComp} [Complementizer]

ComplementVP => TO VP [Infinitive]

<u>WhPhrase rules (3):</u>

WhPhrase => NP {WhElement}

WhPhrase => AdjP {WhElement}

WhPhrase => PP {WhElement}

<u>WhRelPhrase rules (2):</u>

WhRelPhrase => NP {RelElement}

WhRelPhrase => PP {RelElement}

<u>VBar rules (6):</u>

VBar => V

VBar => V VBar {PossibleAuxiliary} [Auxiliary]

VBar => V NEG VBar {PossibleAuxiliary} [Auxiliary] [Negation]

VBar => Modal VBar [Auxiliary]

VBar => Modal NEG VBar [Auxiliary] [Negation]

VBar => AdvP VBar [AdverbialModifier]

<u>VP rules (10):</u>

VP => VBar

VP => VP NP {Copula} [PredicateNominal]

VP => VP AdjP {Copula} [PredicateAdjective]

VP => VP PP {Copula} [PredicatePP]

VP => VP NP {Transitive} [DirectObject]

VP => VP AdjP {TakesAdjectiveComplement} [AdjectiveComplement]

VP => VP PP {TakesPPComplement} [PPComplement]

VP => VP ComplementS {TakesClause} [ClausalComplement]

VP => VP NP {TakesClause} [QuestionOnClausalComplement]

VP => VP AdvP [AdverbialModifier]

<u>NBar rules (6):</u>

NBar => N

NBar => N NBar {PossibleNounModifier} [NounModifier]

NBar => N NBar {PossiblePreName} [PreNameModifier]

NBar => N Conj N {ProperNoun} [AmpersandConjuntion] [ProperNameWithAmpersand]

NBar => V NBar {PossibleVerbalModifier} [VerbalModifier]

NBar => AdjBar NBar [AdjectiveModifier]

<u>CoreNP rules (5):</u>

CoreNP => NBar

CoreNP => NBar NBar {TitleNoun} [TitleModifier]

CoreNP => NBar Identifier {PossiblePreName} [IdentifierModifier]

CoreNP => Title NBar [TitleModifier]

CoreNP => DeterminerGroup NBar [NPSpec]

<u>DeterminerGroup rules (4):</u>

DeterminerGroup => Num

DeterminerGroup => Determiner

DeterminerGroup => NP POS [PossessiveHead]

DeterminerGroup => PreDet Determiner [Predeterminer]

NP rules (6):

NP => Pronoun

NP => CoreNP

NP => NP PP {TakesPPComplement} [PPComplement]

NP => NP ComplementS {TakesClause} [ClausalComplement]

NP => NP NPAppositive [AppositiveModifier]

NP => NP NumAppositive [NumAppositiveModifier]

NPAppositive rules (1):

NPAppositive => Punctuation NP Punctuation {AppositivePunctuation} [AppositiveSemantics] [ClosingAppositivePunctuation]

NumAppositive rules (1):

NumAppositive => Punctuation Num Punctuation {AppositivePunctuation} [AppositiveSemantics] [ClosingAppositivePunctuation]

AdjBar rules (3):

AdjBar => Adj

AdjBar => NBar AdjBar [NounModifierToAdjective]

AdjBar => NBar Punctuation AdjBar [NounLocationModifierToAdjective] [LocationPunctuation]

AdjP rules (2):

AdjP => AdjBar

AdjP => AdjP PP {TakesPPComplement} [PPComplement]

AdvP rules (4):

AdvP => Adv

AdvP => DayOfWeek

AdvP => Prep DayOfWeek

AdvP => ReportingAdvP Pronoun

<u>PP rules (1):</u>

PP => Prep NP [PPObject]

<u>DatePhrase rules (1):</u>

DatePhrase => MonthDay

<u>MonthDay rules (4):</u>

MonthDay => Num MONTH

MonthDay => MONTH Num

MonthDay => MONTH ORD

MonthDay => ORD MONTH

<u>ReportingAdvP rules (2):</u>

ReportingAdvP => V {SayingVerb}

ReportingAdvP => V AdvP {SayingVerb}

In specific embodiment, the present invention uses a combination of syntactic and semantic composition. As noted, the previous portions of the specification described how the interactions of objects of the classes Interpreter, Edge, GrammarRule, and Constituent produce syntactic interpret structures. This description was simplified in one respect, however: GrammarRules, and the Edges that are associated with them, do not merely check to determine whether the constituent associated with a candidate Edge matches the syntactic category specified in a rule: they also check for the eligibility of the candidate in terms of more fine-grained syntactic and/or semantic information. Moreover, Edges create not only new syntactic constituents from the information contained in grammar rules, they also compose the semantics of the syntactic dependents of that constituent, to form a new semantic object that is the associated meaning representation of the newly constructed constituent.

The finer grained syntactic and semantic well-formedness conditions can be expressed in the form of Roles,e.g., [DirectObject], which are pieces of code associated with GrammarRule objects. For example, the rule that constructs a Sentence out of a NounPhrase and a VerbPhrase, has an associated Role named Subject that allows the Sentence to be constructed only if:

(1) the semantics associated with the VerbPhrase has not yet filled its Subject argument position; and

(2) the semantics associated with the candidate NounPhrase is compatible with the semantic type requirement imposed on the Subject argument by the VerbPhrase semantics (e.g. the semantics of the verb 'sell' requires that its Subject be either a Person or an Organization).

If these conditions are met, the semantics of the subject NounPhrase is bound to the Subject position of the semantics of the VerbPhrase, and the new semantic object formed by binding this argument position is passed to the Sentence that is created as its semantics.
Roles can be represented in rules, for example, as simple symbolic names. These names are often associated with the actual code that is used to perform the type checking and semantic composition by lookup in a table. This allows the code for the same named Role to be used in multiple rules. For example, the Subject role will appear in main clause, complement clauses, relative clauses, declarative sentences, questions, etc.

Semantic representations can be constructed during the course of interpreting. The semantic representations associated with phrases and clauses are created by the interpreting process, by means of the composition of the semantics of dependent constituents; e.g., a sentence gets its semantics by the composition of the semantics of its subject NounPhrase and main predicate VerbPhrase; a transitive VerbPhrase gets its semantics from the composition of the semantics of its head Verb and its NounPhrase direct object, etc. PreTerminals (i.e. the constituents corresponding to the actual words in the sentence, such as Noun, Verb, Adjective) get their semantics either by lexical lookup or by default. When a new, fully specified PreTerminal object, such as a Noun, is formed from an underspecified Constituent, the first thing it does is to consult with a LexicalEntry class object of the appropriate type (e.g. NounEntry for Noun, VerbEntry for Verb, etc.) to determine if its stem has an associated lexical entry in that category. If it does, the PreTerminal object uses the semantic information in the LexicalEntry to create its associated semantic representation object. If it does not, the PreTerminal creates a default semantic representation appropriate for its syntactic category; e.g. a Verb creates a default semantic representation whose semantic type is Event, the least specific type of activity; a Noun creates a default semantic representation whose semantic type is TopType, the least specific semantic type for persons, places, things, or concepts, etc.

In a specific embodiment, the Interpreting module (step 111) also uses a sub-module for identification of semantic types associated with words and phrases (step 117). The present invention uses this sub-module 117 that contains the core conceptual knowledge for the system. The sub-module 117 provides a structure for a semantic type system 113 (resource B), which underlies the processes of lexical inference. The type system 113 is structured along multiple dimensions, where each dimension corresponds to a different aspect of word meaning. As a result each dimension involves a different way of understanding a given entity in the domain and thus corresponds to a different set of questions (i.e. queries) concerning that entity.

These different aspects of word meaning are expressed by means of qualia structure, namely "modes of understanding" of an entity. This is described in J. Pustejovsky, "The Generative Lexicon", MIT Press, 1995, which is herein incorporated by reference. A structured conceptual type involving qualia roles may be defined relative to the following four <u>qualia roles</u>:

<u>formal</u>: the kind of entity
<u>constitutive</u>: the mode of individuation of that entity
<u>telic</u>: the purpose or function of the entity
<u>agentive</u>: how the entity comes into being

Qualia roles can provide building blocks for structuring a concept, such that the types in our type system differ in terms of their internal complexity. Thus, concepts are not organized in terms of a taxonomic ISA link uniquely. EachRather, each conceptual type in the type system is a data structure that incorporates the set of inferences that are available as well as the relations between that type and other entities.

In a specific embodiment, the present invention provides types as identifiers for other types. Here, types are not merely used for structuring information for instances of each particular type, but they also play a crucial role in identifying other types in the text. In other words, the structure of the typing information drives part of the knowledge acquisition process. For example, the information that is associated with a noun denoting a professional role, i.e. “doctor”, permits identification of the type of the associated institution, namely a hospital or a health clinic. Similarly, given the semantics of certain head nouns, it is possible to type their modifiers in compound nominal constructions. For example the noun “"maker”" becomes a fairly reliable identifier of entities that are typed as "products", as in “software maker”.

The present invention also provides a qualia structure as a basis for acquiring knowledge. The qualia structure of a lexical item can be identified in terms of patterns occurring in the text. Each qualia represents a well defined component of meaning which is talked about in texts as well as in people's conversations. For instance, an object which occurs with the predicate "manufacture"---as its direct object--- is understood as an artifactual entity which is brought about through a manufacturing process. This information is exploited to acquire information concerning the AGENTIVE role of a given entity. This strategy is extensible at different levels of specificity, in a way that allows the engine to acquire information concerning the relationship between Microsoft and Windows 98. Similarly, there are grammatical constructs such as "used for", "used in", "good for" which provide reliable identifiers for the TELIC of an entity. There are also patterns that indicate the CONSTITUTIVE information: for companies, expressions involving headquarters, location, and address, all provide the basis for specifying the constitutive aspect; for products, constructions such as "made of" or "made from" indicate the specific components of an entity.

Patterns that identify information concerning the TELIC of an entity, can be useful for simultaneously acquiring the same information about another entity from a different perspective. For example, while we are assigning Microsoft as the entity that specifies the AGENTIVE role of one of its products, we are also building that information as part of Microsoft's TELIC.

Qualia Structure can also be used as a basis for querying and reasoning over a database. Given the close relation between syntactic patterns and the semantics they convey relative to a given entity, the querying exploits the same principles used in the knowledge acquisition. The process, however, is taken further. Once we have available the information concerning the AGENTIVE role of a set of different software products, for instance, then it is possible to ask about competitors. To achieve this it is a matter of finding the list of companies that appear in the AGENTIVE role of products with the same type. Similarly, it is possible to query a set of products or entities that are fulfilling a given function that which is specified in the TELIC role.

According to one example the entire process of populating a database from natural language input in text can be provided, as shown below:

Input document = 0000077400.txt (a Reuters text)

Input sentence (appears at offset position 380 in text):

The S&P500 stock index rose 36.46 points.

Tokenized and tagged:

The/DT S&P500/NNP stock/NN index/NN rose/VBD 36.46/CD points/NNS ./.

Examples of the resources accessed from the lexicon and type system sub-module 113 are shown below:

<u>Lexicon system (Resource A):</u>

```
[VerbEntry
        stem: 'rise'
        type: 'financial rise activity';
        subjectRole: #theme;
        objectRole: #measure
]

[NounEntry
        stem: 'point'
        type:'Measure'
]
```

<u>Type system (Resource B):</u>

```
[GLEventType
  name: 'financial rise activity'
```

```
        formal: #([[rise activity]])
        argumentStructure:
            theme: [[Abstract Object]]
            externalArgument: [[Measure]]
       ]

       [GLType                     //comment: Qualia Role
        name: 'Measure'
        formal: #([[Abstract Object]])
]
```

The resulting semantic root node for this text is as follows: :

```
       [UtteranceLexLF
        type: [[Opinion]]
        illocutionaryForce: #Assertion
        content: [FunctionLexLF
                   type: [[rise activity]]
                   predicateStem: 'rise'
                   complements: (#Subject -> [EntityLexLF
                                        type: [[Abstract Object.Company]]
                                        value: 'S&P500 stock index'
                                        quantification: [QuantifierLexLF
                                                 type: [[Abstract Object]]
                                                 value: 'The']]
                                 #DirectObject -> [EntityLexLF
                                                 type: [[Measure]]
                                                 value: 'points'
                                                 quantification: [CountLexLF
                                                          type: [[Number]]
                                                          value: 36.46]])]]
```

Next, a specific embodiment of the present method uses a step of extracting objects, step 125. The present method creates a semantic representation of objects, during interpreting, including syntactic-semantic composition, and qualia induction, that serve as an interface to a relational database model. Since these semantic representation objects are a combination of data and procedures, similar to objects in an object-oriented systems, they implement their own procedures (i.e., methods) to translate themselves from their object representation into SQL statements that map to a relational database. While there are several classes of semantic representation objects (LexLF) that are used during the course of interpreting and semantic composition, there are two classes that are relevant to interactions with the relational database:

EntityLexLF --- which represents the semantics of entities (i.e. person, places, things, concepts)

FunctionLexLF --- which represents the semantics of relations among entities

Both of these classes implement methods for two modes of database interaction: insertion and retrieval.

In insertion mode, the LexLF transforms its semantics into a SQL INSERT statement to add information to the database. Since a LexLF contains various pieces of information that must be present in the database, the LexLF negotiates with the database to discover whether supporting pieces of information exist already or not. For example, FunctionLexLF contains information about the predicate type of the associated predicate and about the various entities that fill its argument positions. Each of these elements must exist in the appropriate table of the database; when one of these elements already exists, the LexLF merely looks up the primary key for it; when it does not, the LexLF must first cause the element to be inserted in the database, to preserve relational integrity.

The entities are inserted, then the relation is built on them.

The EntityLexLF for 'S&P500 stock index' produces the following SQL statements:

Insert the entity:

```
insert into Entities(CanonicalName) values('S&P500 stock index')
```

Then insert the type information:

(autonumbering in database at time of insertion gives 'S&P500 stock index' an EntityID of 5230)

```
insert into Types(EntityID, DocumentID, Offset, Type)
        values(5230,405,380,'Abstract Object.Company')
```

The EntityLexLF for '36.46 points' produces the following:

```
insert into Entities(CanonicalName) values('point')
insert into Types(EntityID, DocumentID, Offset, Type)
        values(5231,405,380,'Measure')
```

The FunctionLexLF for the whole utterance produces the following:

Insert the predicate:

```
insert into Predicates(PredicateName, PredicateType )
        values('rise-rise activity','rise activity')
```

Then the relation proper:

(the EntityId for unfilled arguments is '0' by default)

```
insert into Relations(PredicateID,DocumentID,Offset,Subject,Object,
        ClausalArg,ExtraArg1, ExtraArg2,ExtraArg3)

values(23,405,380,5230,5231,0,0,0,0)
```

Insert the cardinality information from the Direct Object

```
update Relations set ObjectCardinality = '36.46' where RelationID = 776
```

Once the objects have been extracted, step 125, they can be mapped onto a relational database, which is illustrated by a simplified method diagram of Fig. 2. This diagram is merely an example and should not limit the scope of the claims herein. One of ordinary skill in the art would recognize other variations, modifications, and alternatives. The present method 200 begins at step 201, which can derive from step 125 in the above Fig. The method maps (step 203) objects in a relational manner, which can be defined by an entity relation diagram, for example. The objects are inserted (step 205) into an objected oriented relational database 211. Once the objects have been inserted they are now ready for use by a user.

In a specific embodiment, the user can query (step 207) the database 211. The present process stops at step 209, but is not limited. As merely an example, which uses the above database population technique, it would be possible to find an answer to the following question:

What did the S&P stock index do?

As in the previous example, this utterance would go through the stages of at least tagging and tokenization:

What/WP did/VBD the/DT S&P500/NNP stock/NN index/NN do/VB ?/.

and would produce a semantic representation of the following form:

```
[UtteranceLexLF
 type: [[Question]]
 illocutionaryForce: #WhQuestion
 content: [FunctionLexLF
        type: [[QueryDo]]
        predicateStem: 'do'
        complements: (#Subject -> [EntityLexLF
                           type: [[Abstract Object]]
                           value: 'S&P500 stock index'
                           quantification: [QuantifierLexLF
                                    type: [[Abstract Object]]
                                    value: 'The']]
                      #DirectObject -> [EntityLexLF
                              type: [[Entity]]
                              value: 'What'
                              quantification: [QuantifierLexLF
                                       type: [[Entity]]
                                       value: 'what'
                                       quantifier: #Wh]])]]
```

There are several features of this semantic form. First, the semantics of the interrogative pronoun 'What' is interpreted in its 'logical' position, i.e. as the direct object of the main verb 'do'. Second, the semantic representation of 'What' includes a QuantifierLexLF that has #Wh as the value of its #quantifier. This indicates that this is the logical argument that is being asked about in this query.

Semantic representations for content queries of this type are processed for database lookup in the following manner.

First, the EntityID of the subject is retrieved:

```
select EntityID from Entities where CanonicalName = 'S&P500 stock
        index'
```

This will retrieve the EntityID 5230, which is then used to construct a select statement on the Relations table:

```
select * from Relations where Subject = 5230
```

This will retrieve the row:

```
(776,23,405,380,5230,null,5231,'36.46',0,0,null,0,null,0,null,0)
```

Finally, for presentation to the user, the system will use this information to retrieve the sentence:

The S&P500 stock index rose 36.46 points.

i.e. the sentence at offset position 380, in the document with DocumentID 405, whose filename is '0000077400'. In the current implementation, this information is passed to the user interface in the format:

```
<DISPLAY-FULL-OBJECT ""
{ "Reuters"
  "http://199.103.231.59/demo-
        code/source.pl/display=0000077400,380#380"
  "The S&P500 stock index rose 36.46 points." }  { } >
```

which contains the source of the response text, a URL that points to the complete source document (the location of the host machine is not hard-coded, but is determined dynamically to point to the host machine that is running the system), and the actual response text. This is presented in a more user-friendly format: only the name of the source and the response text are displayed, with the source name being a hyperlink that points to the full source text, to allow the user to examine the entire text, if desired.

Note that the database lookup is done in two stages: an initial lookup of the EntityID of the subject and then a lookup from the Relations table, using this EntityID, rather than as a single select statement that does a database join over these two tables. The reason for the decomposition of the lookup process in this manner is to allow for a more responsive interaction with the end user. Ifa simple join query fails to retrieve any data, the source of the failure is unknown. It could be due to, for example, one of several reasons:

1. one or more of the entities asked about are unknown to the system;

2. the entities are known, and appear as arguments to one or more relations, but not as arguments to the relation the user specifically asked about;

3. the types of the entities are known, but there is no relation that connects them;

4. etc.

By decomposing the lookup process into separate stages, the system can explore the possibility of presenting the user with a response that is more informative than a simple 'No Answer'. It can provide the user with an exact answer if one is available, and with the closest appropriate answer where an exact match is impossible. 'No Answer' responses are only given when the system cannot find anything that is relevant to the user's question.

Using the above example in retrieval (e.g., query) mode, the LexLF transforms its semantics into a SQL SELECT statement. This procedure is the mirror image of insertion in many ways: elements from multiple tables should be queried to produce the desired answer. Moreover, to provide robustness in the system's replies to the user, the LexLF may engage in iterative interactions with the database, as it falls back and widens the parameters of its retrieval space to find information relevant to the user's request, but which may not literally match the query. For example, if the user asks 'Did Brand X buy The Highpriced Spread?', the database may not contain information about the buy relation between Brand X and The Highpriced Spread, but it may contain other relations involving them. The LexLF will engage in a series of transactions with the database to find something at least loosely matching the user's search parameters before giving up and returning no information. There are two sorts of transactions possible with the database, when there is no direct answer to the query:

a. If there are other relations, in which the object being queried about appears, they will be presented, when related to the original query relation.
b. If there are more general relations or object type descriptions than those present in the query, these will be returned.

Although the above functionality has generally been described in terms of specific hardware and software, it would be recognized that the invention has a much broader range of applicability. For example, the software functionality can be further combined or even separated. Similarly, the hardware functionality can be further combined, or even separated. The software functionality can be implemented in terms of hardware or a combination of hardware and software. Similarly, the hardware functionality can be implemented in software or a combination of hardware and software. Any number of different combinations can occur depending upon the application.

Many modifications and variations of the present invention are possible in light of the above teachings. Therefore, it is to be understood that within the scope of the appended claims, the invention may be practiced otherwise than as specifically described.

WHAT IS CLAIMED IS:

1. A natural language database forming method, said method comprising:

providing text information comprising a plurality of related words;

tagging each word in said text information;

forming an object comprising syntactic information and semantic information from each word in said text information; and

placing into a object oriented relational database said object.

2. The method of claim 1 wherein said tag can be selected from a verb, a noun, an adjective, an adverb, a numeral, a conjunction, a determiner, and a preposition.

3. The method of claim 1 wherein said text information is derived from publications, e-mail, newspapers, news feeds, and wires.

4. The method of claim 1 wherein said relational object oriented database is a mixed objected oriented database.

5. The method of claim 1 further comprising:

forming a query; and

selecting an object based upon an entity relationships to achieve a unique output.

6. The method of claim 5 wherein said output comprises text information.

7. The method of claim 5 wherein said unique output comprises an answer.

8. The method of claim 6 wherein said text information supports the answer.

9. The method of claim 5 wherein is said text information comprises a plurality of headings.

10. The method of claim 1 wherein said text information is provided in electronic form.

11 . A natural language knowledge acquisition method, said method comprising:

providing text information comprising a plurality of related words, said text information being in electronic form;

tagging each word in said text information;

forming an object comprising syntactic information and semantic information from each word in said text information;

placing into a object oriented relational database said object;

repeating said providing, tagging, forming, and placing to populate said relational object oriented database;

forming a query; and

selecting an object based upon an entity relationships to achieve a unique output.

12. The method of claim 11 wherein said tag can be selected from a . verb, a noun, an adjective, an adverb, a numeral, a connection, a determiner, and a preposition.

13. The method of claim 11 wherein said text information is derived from publications, e-mail, newspapers, news feeds, and wires.

14. The method of claim 11 wherein said relational object oriented database is a mixed objected oriented database.

15. The method of claim 11 wherein said output comprises text information.

16. The method of claim 11 wherein said unique output comprises an answer.

17. The method of claim 11 wherein said text information supports the answer.

18. The method of claim 11 wherein is said text information comprises a plurality of headings.

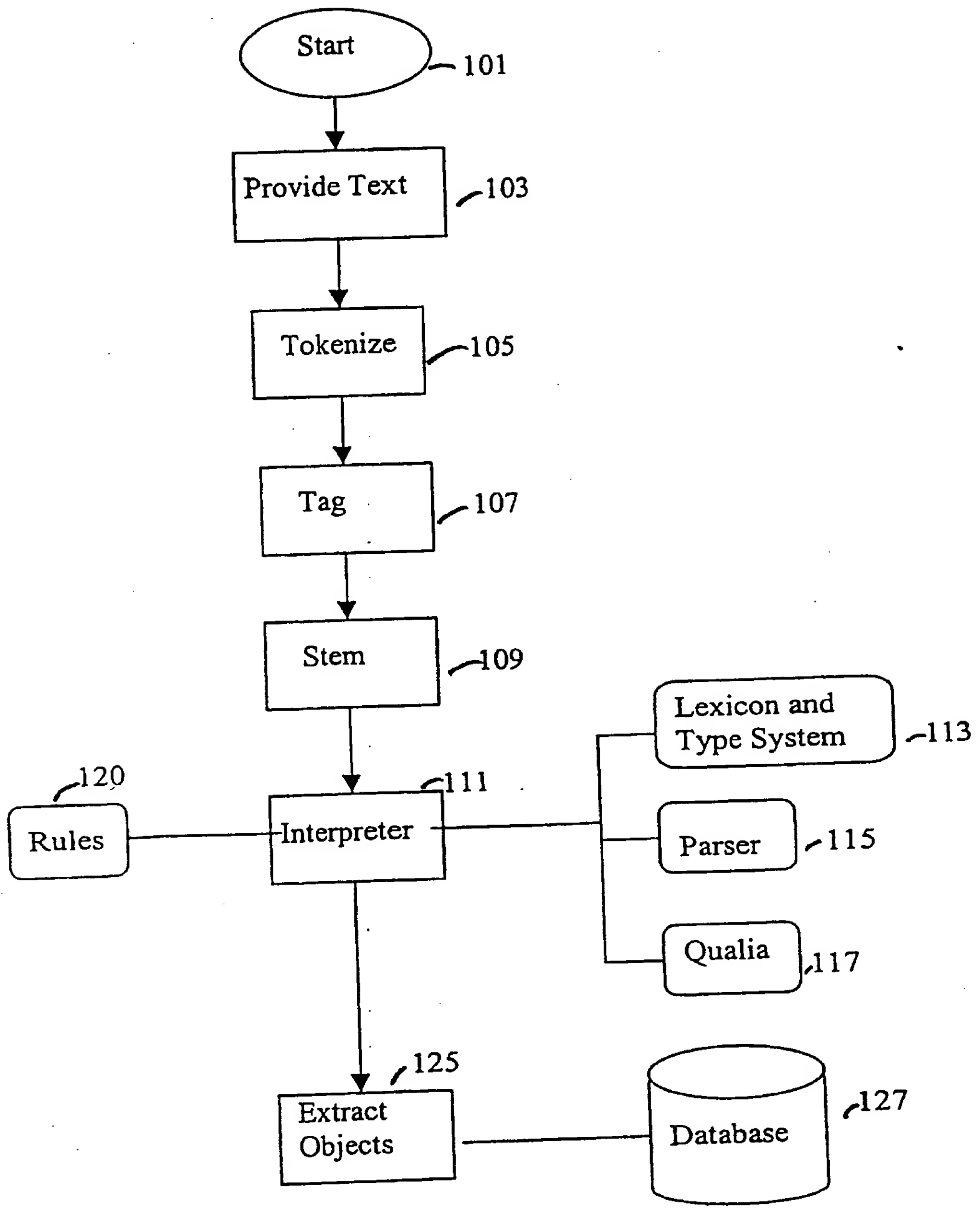

Start
101
Provide Text
103
Tokenize
105
Tag
107
Stem
109
Lexicon and Type System
113
120
Rules
Interpreter
111
Parser
115
Qualia
117
125
Extract Objects
Database
127

FIG. 1

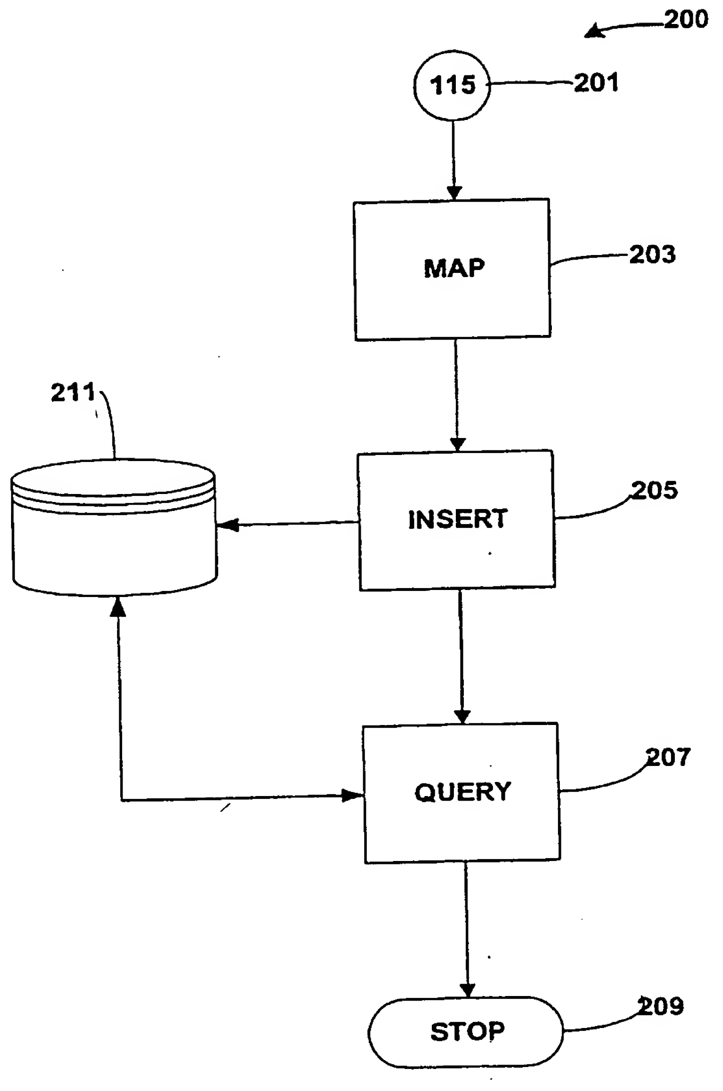


FIG. 2

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US99/28226

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) :GO6F 17/30
US CL :707/1, 3, 4, 5
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 707/1, 3, 4, 5

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

West, CAS Online

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y, P	US 5,909,678 A (BERGMAN et al.) 01 June 1999, see abstract.	1-18
Y	US 5,584,024 A (SHWARTZ) 10 December 1996, See figs 11A & 11B.	1-18
A	US 4,688,195 A (THOMPSON et al.) 18 August 1987, See abstract.	1, 11
A	US 4,829,423 A (TENNANT et al.) 09 May 1989, col. 6, lines 25-35.	1, 11
A	US 5,555,367 (PREMERLANI et al.) 10 September 1996, See abstract.	1, 11

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:
"A"	document defining the general state of the art which is not considered to be of particular relevance
"E"	earlier document published on or after the international filing date
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O"	document referring to an oral disclosure, use, exhibition or other means
"P"	document published prior to the international filing date but later than the priority date claimed
"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&"	document member of the same patent family

Date of the actual completion of the international search
11 MARCH 2000

Date of mailing of the international search report
12 APR 2000

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231
Facsimile No. (703) 305-3230

Authorized officer
[illegible]
SANJIV SHAH
Telephone No. (703) 305-8355